Dynamic Complexity: Basics and Recent Directions

Thomas Schwentick

(with some borrowed slides from Nils Vortmeier and Thomas Zeume)

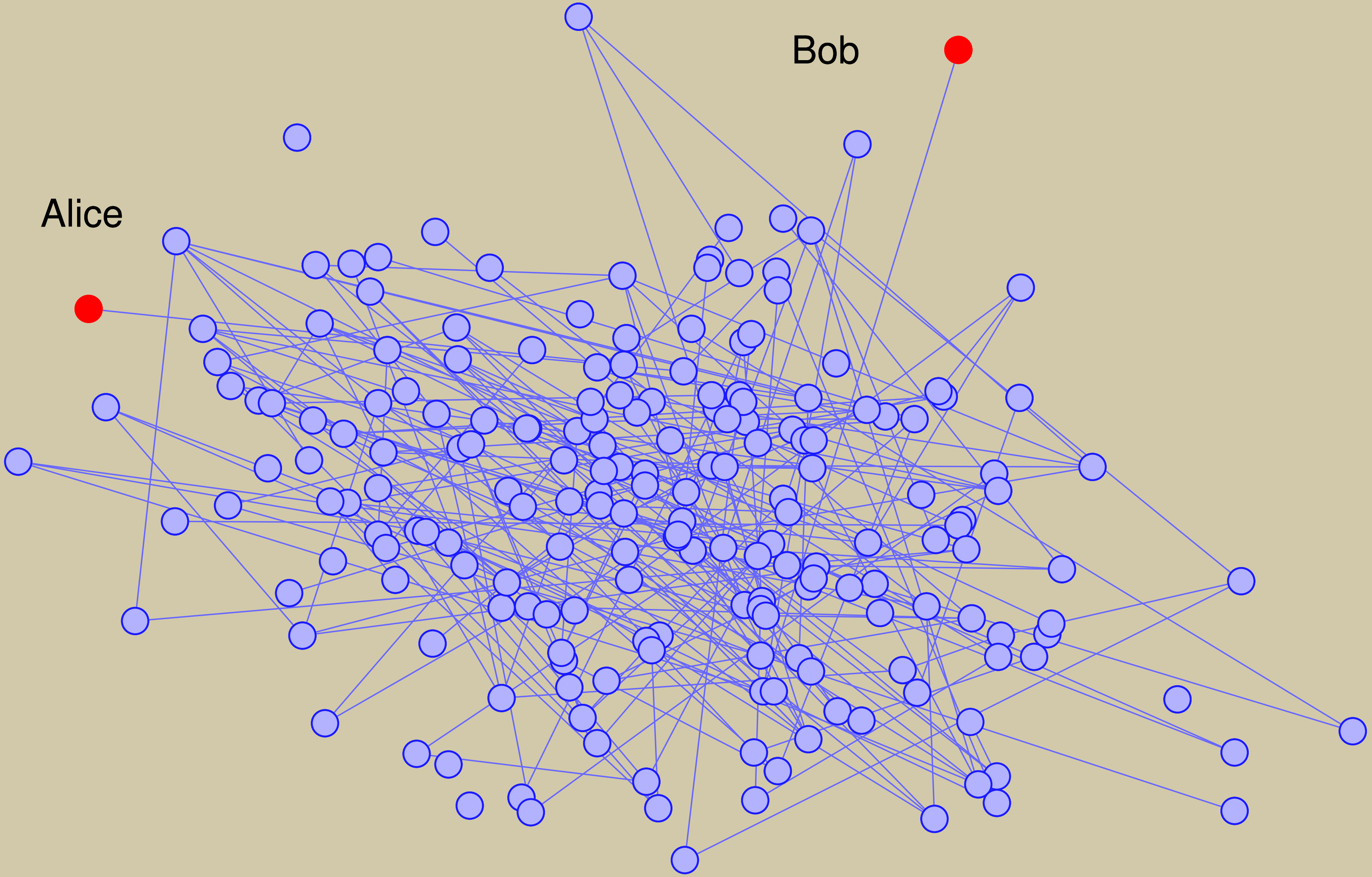
Chennai/Dortmund, March 2021

technische universität dortmund

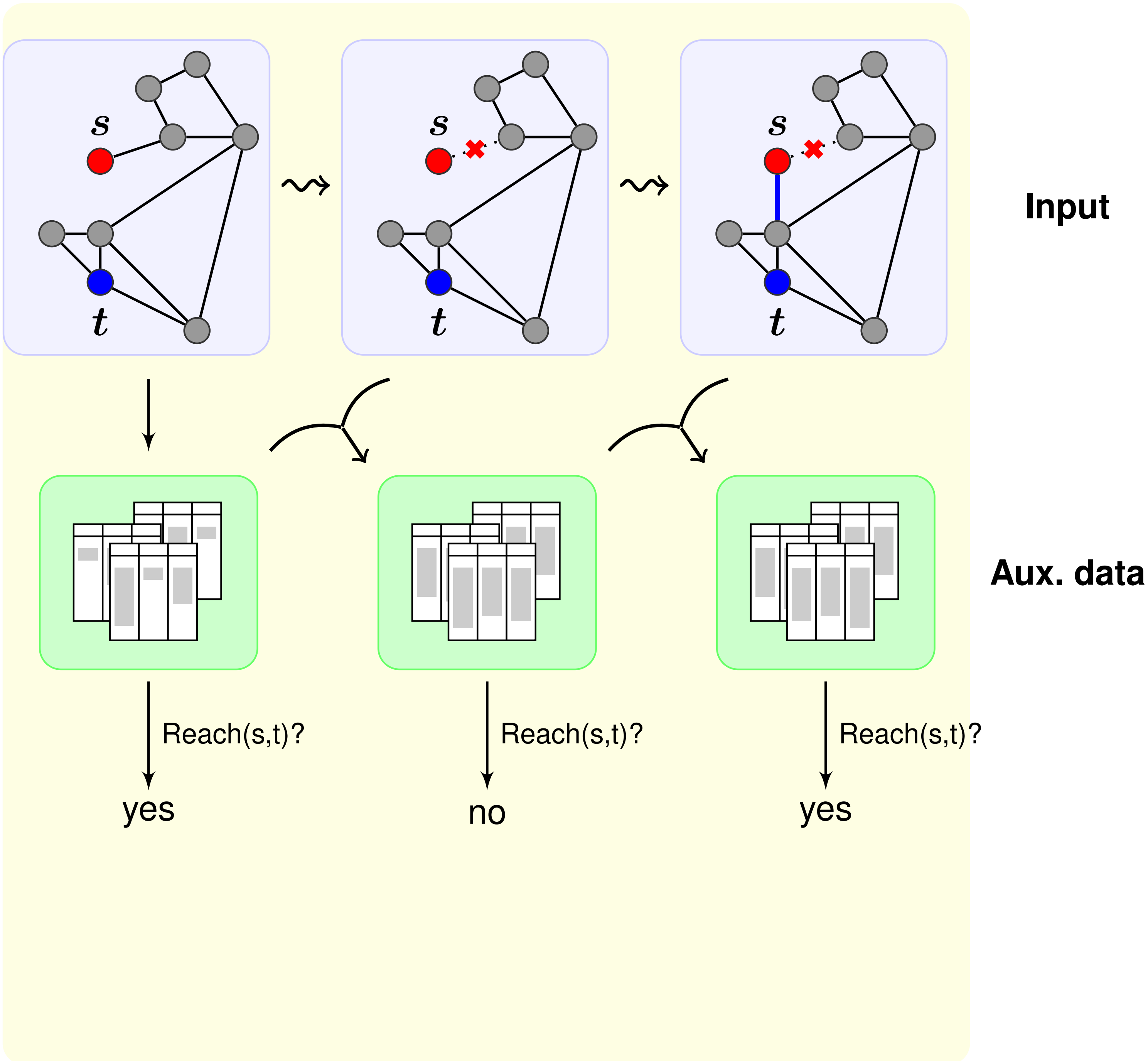
Lehrstuhl Logik in der Informatik

Dynamic Reachability in Practice: Social Networks

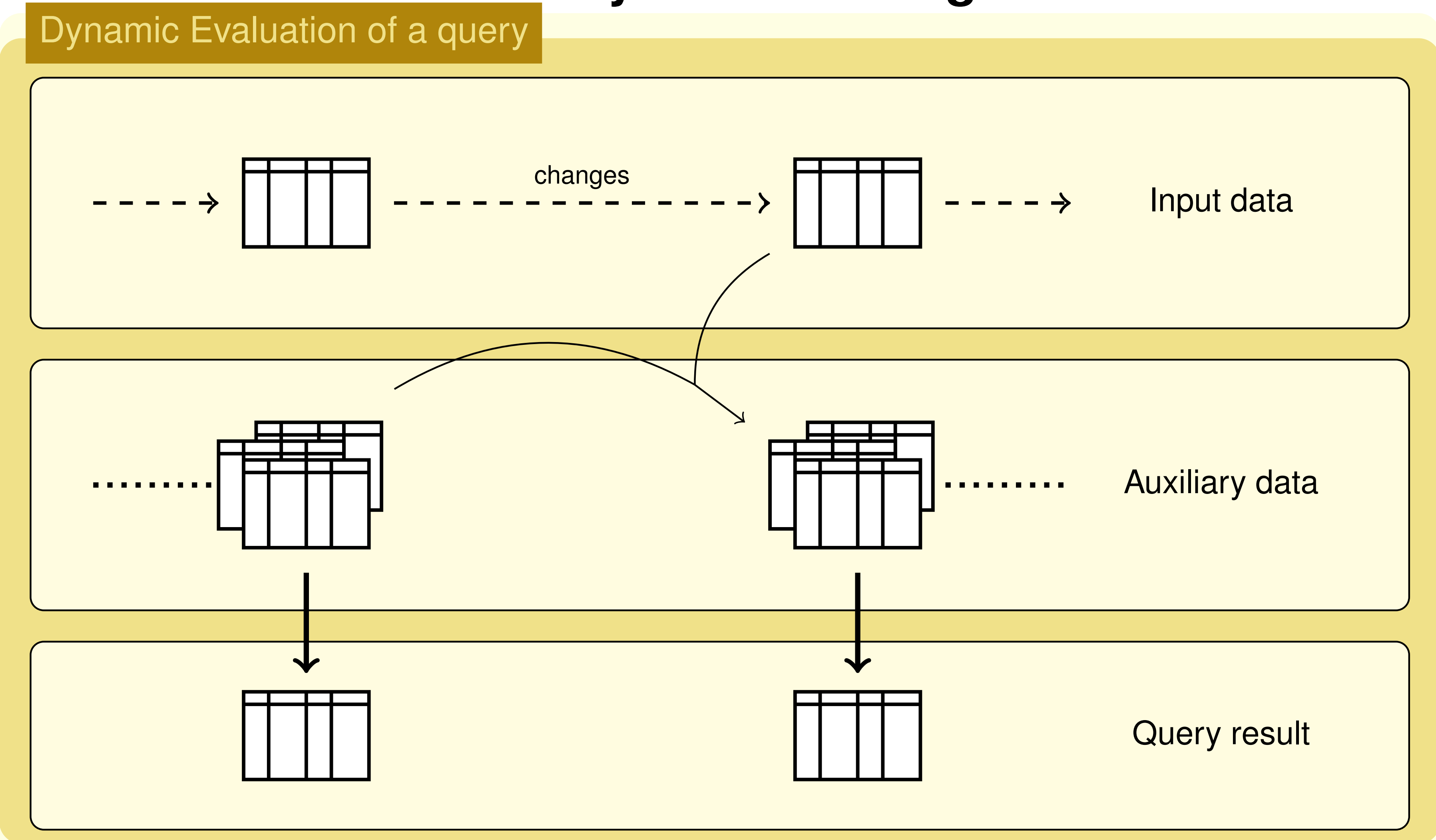
Example



The Dynamic Setting: Reachability



The Dynamic Setting

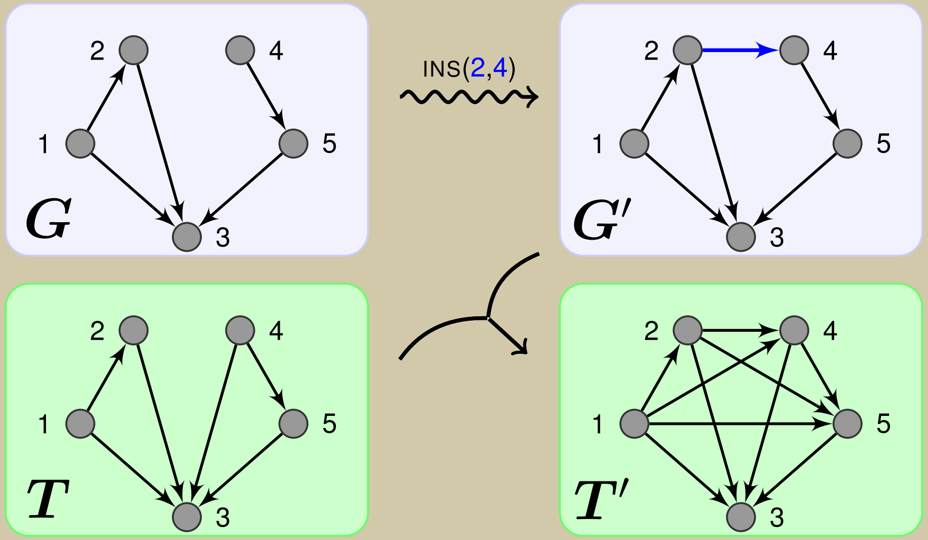


- **DynFO**: Auxiliary relations are updated using first-order logic

...queries "maintainable" by first-order logic

Example 1: Reachability, Insertions only

Example



Simple idea

- Store the transitive closure of the edge relation in a binary auxiliary relation T ☞[Dong, Su 93/95; Patnaik, Immerman 94/97]

Update rule

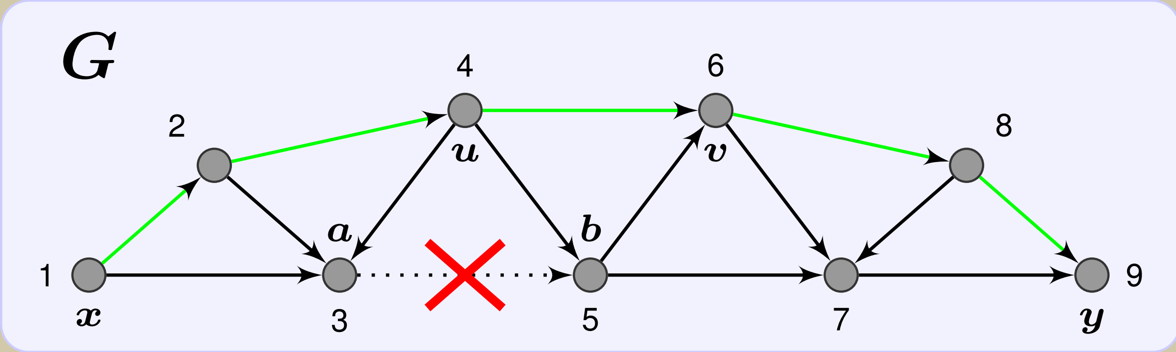
- **on insert (u, v) into E**
  update $T(x, y)$ as $T(x, y) \vee \big(T(x, u) \wedge T(v, y)\big)$

- determines the pairs (x, y) in T after insertion of (u, v) to E

- Transitive closure does not suffice for edge *deletions* [Dong, Libkin, Wong 95]

Example 2: Reachability, with Deletions, Acyclic Graphs Basics

Example



- For *directed acyclic* graphs, Reachability can be maintained with first-order updates [Dong, Su 93/95; Patnaik, Immerman 94/97]

Challenge

- How to express, that there is still a path p from x to y after deleting edge (a, b)?

  Simple cases $E(x, y)$, $\neg T(x, a)$, $\neg T(b, y)$, $\ldots$

  Otherwise p must have a last node $u \neq y$ from which a can be reached

  $$\cdots \vee \exists u, v\big((u \neq a \vee v \neq b) \wedge T(x, u) \wedge E(u, v) \wedge T(v, y) \wedge T(u, a) \wedge \neg T(v, a)\big)$$

Why DynFO?

- A query can be maintained in **DynFO**, if and only if it can be maintained

- by polynomially many parallel processors in time $\mathcal{O}(1)$
- by means of the relational algebra
  ☞ aka core SQL
- in $\mathbf{AC^0}$ ☞ the "lowest complexity class"

Goals of this Research

General goals of our research

- Understand the expressive power of **DynFO**

- Which queries are in **DynFO**?
  - General techniques for **DynFO** programs
- Which queries are **not** in **DynFO**?
  - Methods for inexpressibility results?

Lessons learned:

- In the dynamic setting, first-order logic is much more powerful than in the static setting
- Inexpressibility results are hard to obtain

Dynamic Complexity: Our Setting in More Detail

- Our setting, in a nutshell

Simple change operations

- Insertion of a single tuple: **insert** (u, v)
- Deletion of a single tuple: **delete** (u, v)

Fixed universe

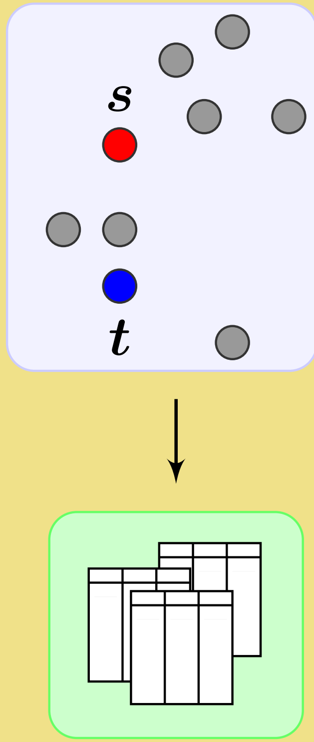
- Set of nodes is fixed, for each computation

☞ n = number of nodes

Dynamic program

- One update formula per change operation and auxiliary relation
- One auxiliary relation yields the output

Initialisation



In principle

- **empty input** and **empty auxiliary data**

- But we can always* assume the nodes are numbers $1, \ldots, n$ and formulas can use

☞ *: almost

  - a linear order $\leqslant$ on the nodes, and
  - addition and multiplication relations

Short History of Dynamic Complexity

2019 — Complex Changes, Muddling, Work

2015 — Reach in non-uniform **DynAC**$^0[\oplus]$; Reach in **DynFO**

2010 — Connections to static complexity classes; Lower bounds for small syntactic fragments

2005 — Reach in dynamic **TC**^{0}

2000 — Maintaining basic graph queries; Lower bounds for restricted Arity

1995

1990 — Introduction of the setting

Contents

Undirected Reachability in DynFO (1/3)

- We already know:

Theorem [Patnaik, Immerman 94/97]

- ACYCLIC REACH ∈ **DynFO**

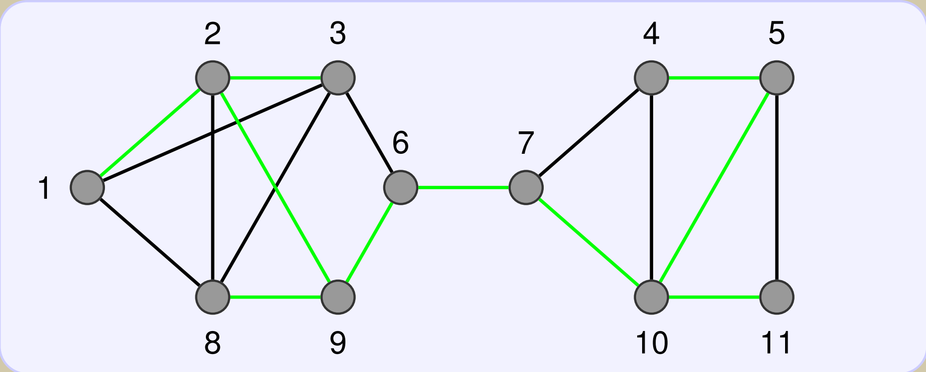
SYM-REACH

- Reachability for undirected graphs

- There are several proofs for SYM-REACH ∈ **DynFO**
- We look at the simplest and first proof by [Patnaik, Immerman 94/97]

Undirected Reachability in DynFO (2/3)

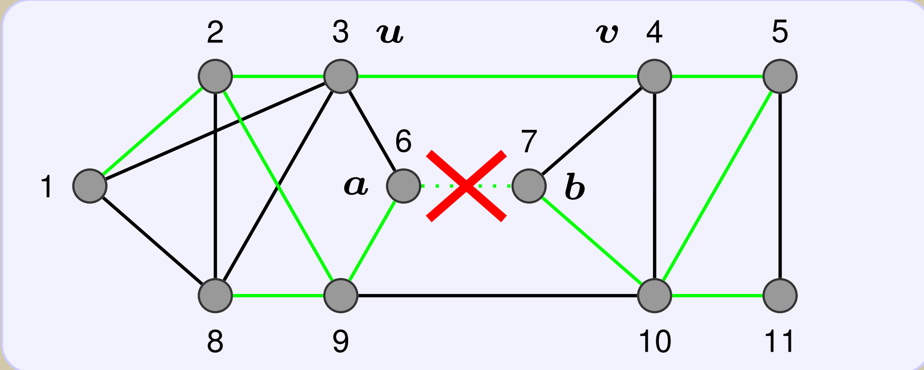
Example: Insertion



Basic idea

- Maintain a spanning forest F and its transitive closure T
- On arrival of a new edge, add it to F, if it connects two distinct components

Example: Deletion



- Deletion is more tricky, again

- How to modify the spanning tree if an edge (a, b) is deleted but its component remains connected?
  - Determine nodes u and v in the subtrees of a and b, respectively, such that $(u, v) \in E$, and add (u, v) to F

- This can be done with
  - a more sophisticated relation T with all triples (d, e, g) for which there is a path in F from d to e through g
  - some order on the edges to choose (u, v) *uniquely*

Undirected Reachability in DynFO (3/3)

Theorem [Patnaik, Immerman 94/97]

- SYM-REACH $\in$ **DynFO**

- Is the ternary auxiliary relation T necessary? ☞ No

k-ary DynFO

- Queries in **DynFO** that can be maintained with (at most) k-ary aux relations

Theorem [Dong, Su 95/98]

- SYM-REACH $\in$ binary **DynFO**
- SYM-REACH $\notin$ unary **DynFO**

Results about Directed Reachability

Conjecture [Patnaik, Immerman 97]

- Reachability is in **DynFO**

Reachability is in DynFO for

- acyclic graphs [Patnaik, Immerman 94/97]
- undirected graphs [Patnaik, Immerman 94/97; Dong, Su 98, Grädel, Siebertz 12]
- embedded planar graphs [Datta, Hesse, Kulkarni 14]

Reachability is in DynFO extended by

- counting quantifiers [Hesse 01]
- modulo-2 counting quantifiers [Datta, Hesse, Kulkarni 14]

Theorem [Datta, Kulkarni, Mukherjee, TS, Zeume 15]

- Reachability is in **DynFO**

Directed Reachability in DynFO: Outline

Definition: REACH

Input: Directed Graph G

Result: All pairs (s, t) for which there is a path from s to t in G

Definition: FULLRANK

Input: $(m \times m)$-matrix A with values from $\{0, \ldots, m\}$

Question: Does A have full rank m?

Definition: FULLRANKMODP

Input: $(m \times m)$-matrix A with values from $\{0, \ldots, m\}$, prime $p \leqslant m^2$

Question: Does A have full rank m over $\mathbb{Z}_p$?

Structure of the proof

- We show:
  - (1) REACH $\leqslant_{\text{bfo-tt}[+,\times]}$ FULLRANK
  - (2) FULLRANK $\leqslant_{\text{bfo-tt}}$ FULLRANKMODP
  - (3) FULLRANKMODP $\in$ **DynFO**$(+, \times)$
  - (4) For domain independent Q: $Q \in$ **DynFO**$(+, \times) \Rightarrow Q \in$ **DynFO**

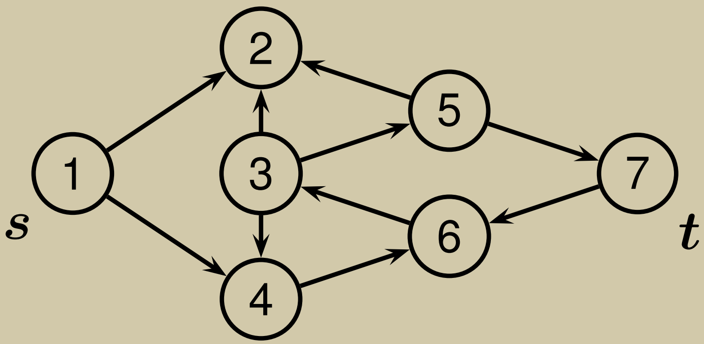
Further ingredients

- **DynFO** is closed under $\leqslant_{\text{bfo-tt}}$-reductions
- **DynFO**$(+, \times)$ is closed under $\leqslant_{\text{bfo-tt}[+,\times]}$-reductions
- REACH is domain independent

- All steps (1)-(4) are relatively simple and build on previous work

Reach vs. FullRank (1/2)

Example



$$A_G = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

Example

$$(A_G)^2 = \begin{pmatrix} 1 & 2 & 0 & 2 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 1 & 2 & 2 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 2 & 0 \\ 0 & 2 & 0 & 0 & 1 & 1 & 2 \\ 0 & 1 & 2 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 2 & 1 \end{pmatrix}$$

$$(A_G)^8 = \begin{pmatrix} \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 57 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \end{pmatrix}$$

Reach vs. FullRank (2/2)

Proof idea

- Let G be a graph with n vertices and A_G its adjacency matrix

- $(A_G)^i[s,t] \neq 0 \iff$ there is a path of length $\leqslant i$ from s to t

Observation (e.g.: Laubner 11)

- $I - \frac{1}{n}A_G$ is invertible and

$$(I - \frac{1}{n}A_G)^{-1} = I + \sum_{i=1}^{\infty} (\frac{1}{n}A_G)^i$$

➡ the s,t-entry of this matrix is zero $\iff$ t is **not** reachable from s

- $B \stackrel{\text{def}}{=} nI - A_G$ ☞ integer matrix

The following are equivalent:

- t is **not** reachable from s
- $B^{-1}[s,t] = 0$

- $$\underbrace{\begin{pmatrix} \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \end{pmatrix}}_{B} \times \underbrace{\begin{pmatrix} x[1] \\ x[s] \\ \cdot \\ \cdot \\ x[n] \end{pmatrix}}_{x} = \underbrace{\begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}}_{e_t}$$

  has a solution with $x[s] = 0$

- $$\underbrace{\begin{pmatrix} \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & B & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & 1 & 0 & 0 & 0 \end{pmatrix}}_{B'} \times \underbrace{\begin{pmatrix} x[1] \\ x[s] \\ \cdot \\ \cdot \\ x[n] \end{pmatrix}}_{x} = \underbrace{\begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}}_{e'_t}$$

  has any solution

A Corollary: Regular Path Queries

Regular Path Queries and Reachability

- Let R be a regular language over Σ
- The regular path query q_R over graph databases asks for all pairs (u, v), for which there is a path from u to v with label sequence in R

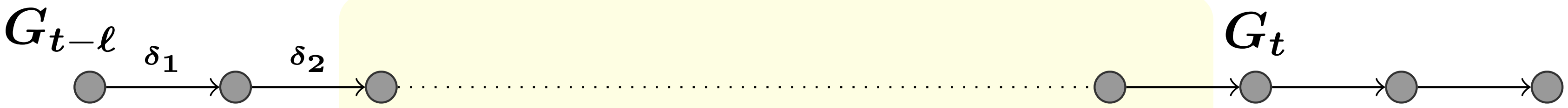
- Let $G = (V, E)$ with edge labels from alphabet Σ
- Let $\mathcal{A}$ be an NFA for R with state set Q and unique initial and final states p_0 and p_f

- Let the product graph $G \times \mathcal{A}$ have
  - node set $V \times Q$,
  - edge $(i, p) \to (j, q)$
    if $i \xrightarrow{\sigma} j$ and $p \xrightarrow{\sigma} q$, for some $\sigma \in \Sigma$

- There is an R-path in G from u to $v \iff$
  (v, p_f) is reachable from (u, p_s) in $G \times \mathcal{A}$
- ... and every change in G yields $\leqslant |Q|$ changes in $G \times \mathcal{A}$ ☞ bfo-reduction

→ Since Directed Reachability is in **DynFO**, Regular Path Queries are in **DynFO** as well

A Useful Technique: Muddling



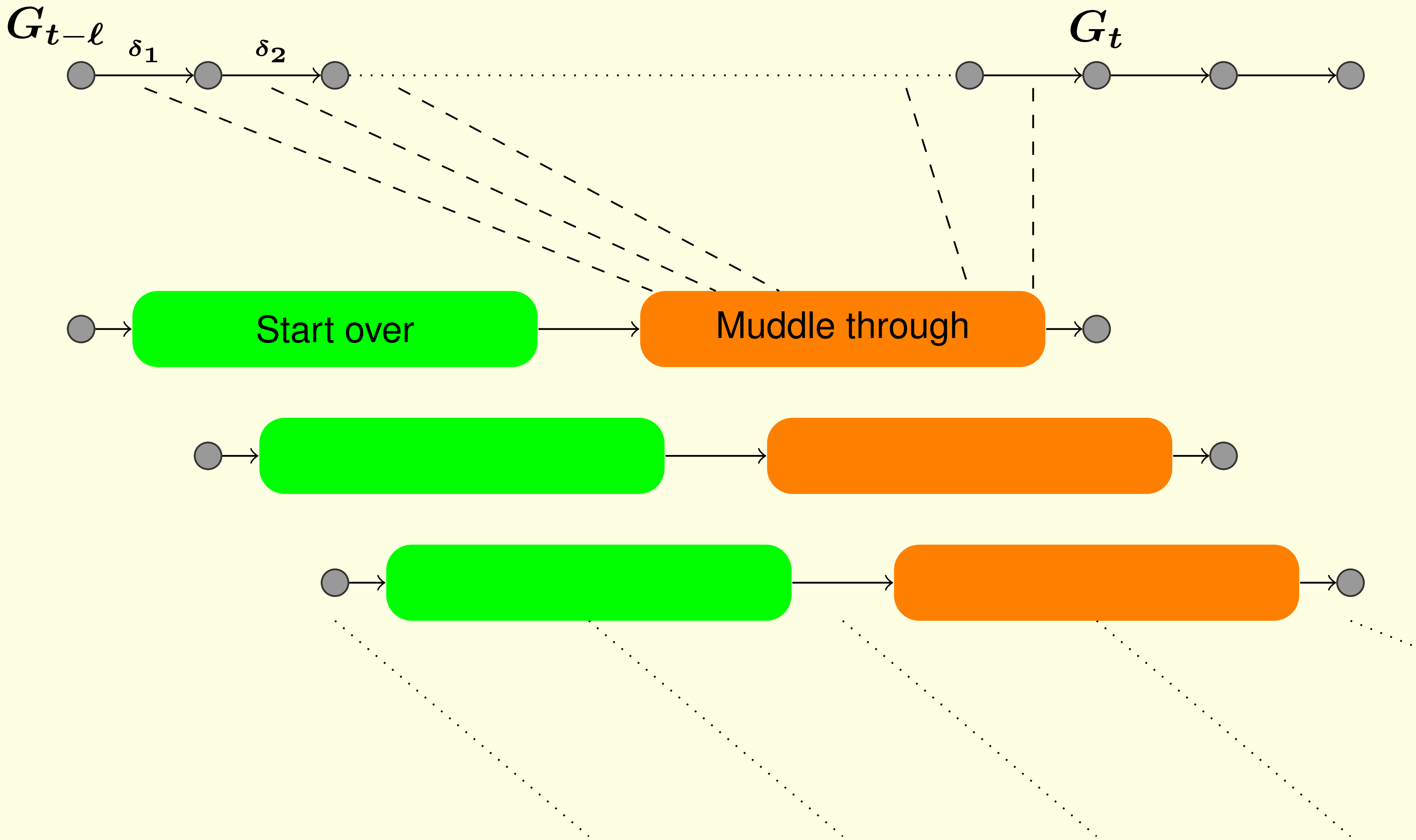
Basic idea of muddling

- To give the correct answer for graph G_t (at time t) do the following:

(1) For a suitable ℓ, start the computation of a solution from scratch at time $t - \ell$ for $G_{t-\ell}$ ☞ Start over

(2) Update the computed solution for the ℓ changes between $t - \ell$ and t with constant speed-up ☞ Muddle through

Muddling: basic idea



Muddling Lemma

Muddling Lemma [Datta, Mukherjee, S., Vortmeier, Zeume 17]

- A query Q is in **DynFO**, if it has the following property:
  - From a graph G of size n
  - ... one can compute auxiliary relations in **AC**^{1} ...
  - ... based on which the query can be **FO**-maintained for $\log n$ change steps

What is AC^1?

- **AC**^{1} is a complexity class based on circuits of logarithmic depth
- **LOGSPACE** $\subseteq$ **NL** $\subseteq$ **AC**^{1}

- It can be characterised in terms of a limited fixed-point process: ☞ Immerman

- **AC**^{1} $=$ **IND**$(\log n)$,
  i.e., all queries that can be evaluated by $\mathcal{O}(\log n)$ many applications of the same **FO**-formula

Example

- $\log n+1$ applications of
  $$\varphi(x,y) = E(x,y) \vee \exists z(T(x,z) \wedge T(z,y))$$
  yield the transitive closure of a graph

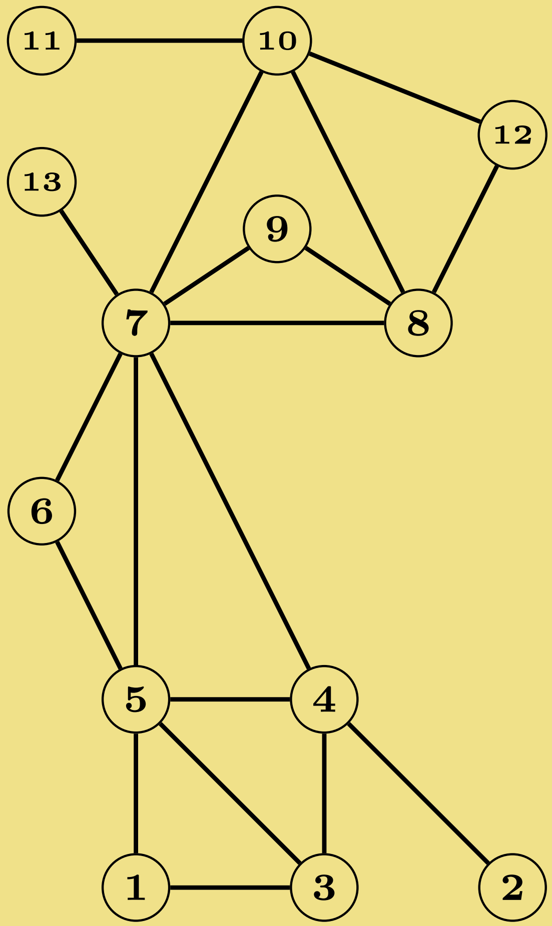
Application: 3-COL on bounded tree-width Graphs

Theorem [Datta, Mukherjee, S., Vortmeier, Zeume 17]

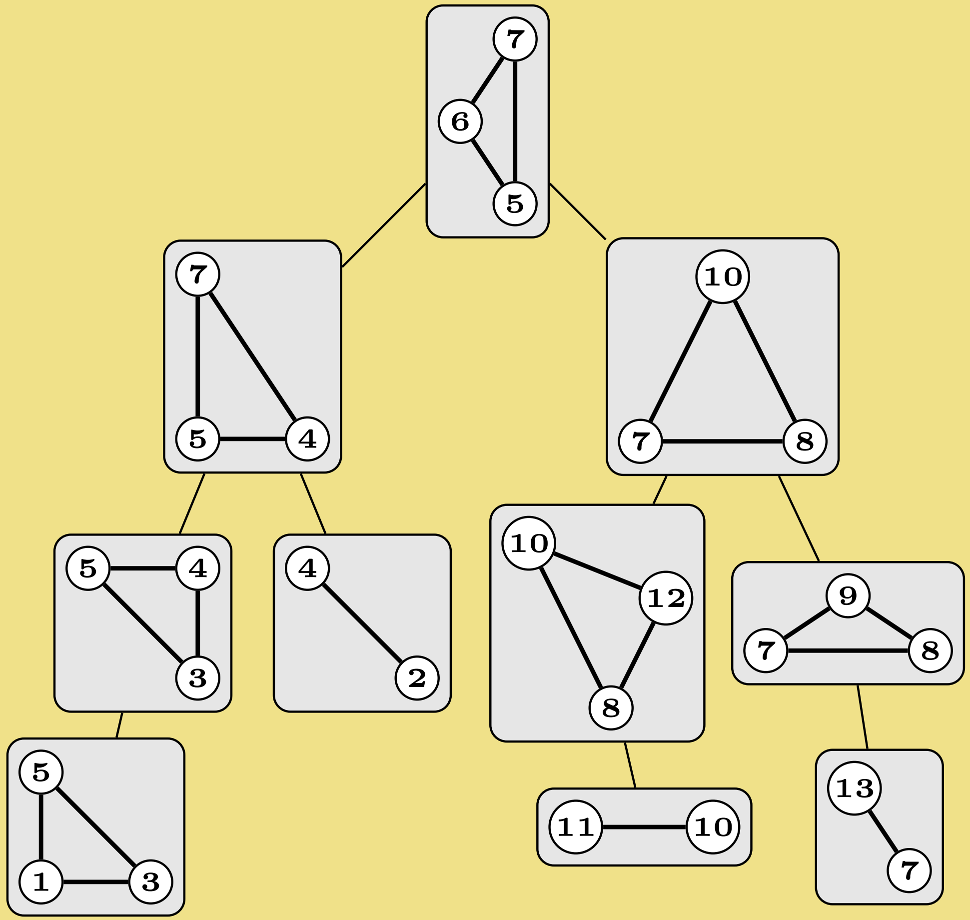
- 3-COL can be maintained in **DynFO** on graphs of bounded tree-width

Tree decompositions

An input graph ...



... and its tree decomposition



Application: 3-COL on bounded tree-width Graphs

Theorem [Datta, Mukherjee, S., Vortmeier, Zeume 17]

- 3-COL can be maintained in **DynFO** on graphs of bounded tree-width

Challenge

- A small change of the graph might induce a big change of the tree decomposition

Muddling can help!

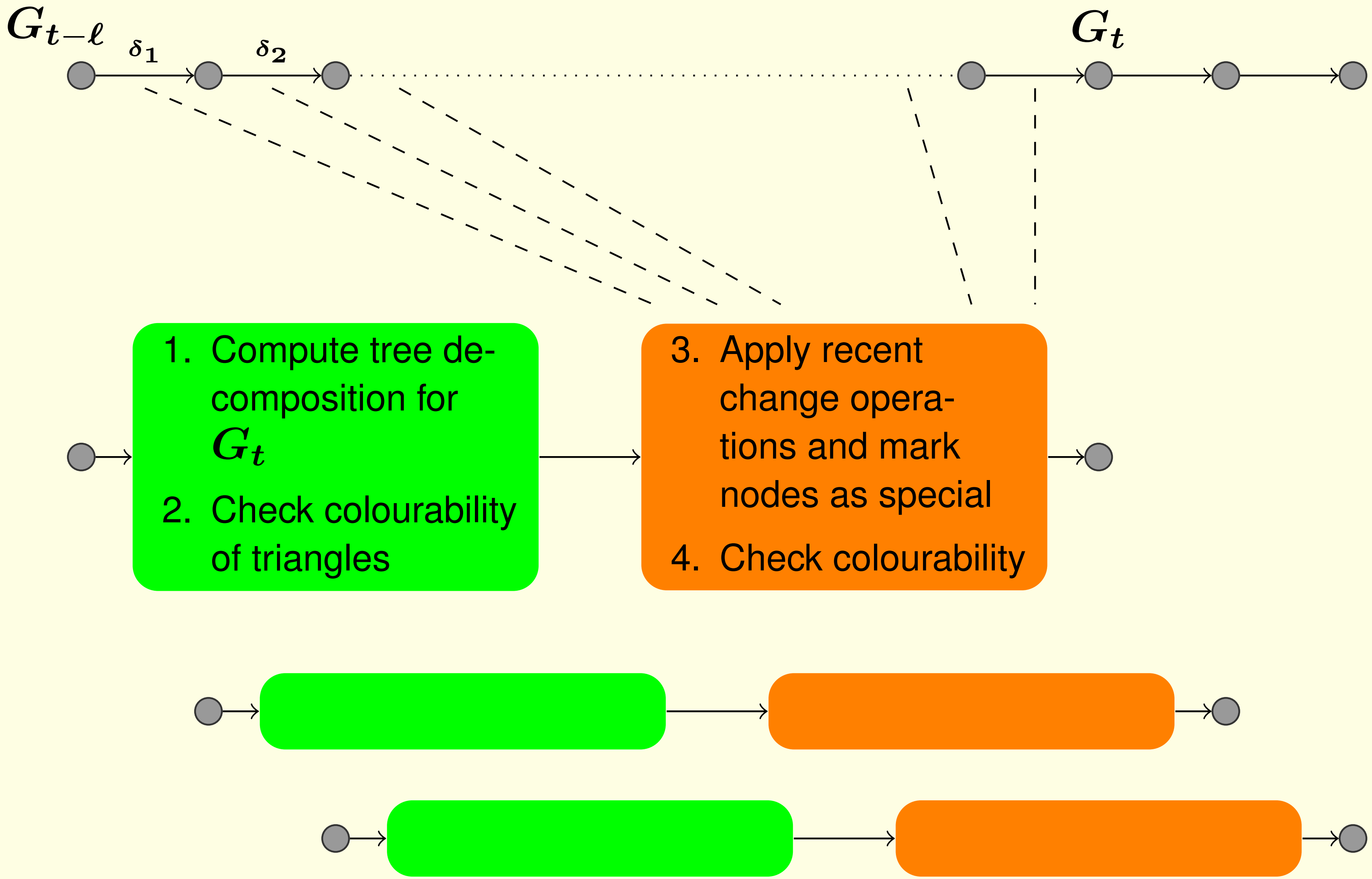
- Tree decompositions can be computed in logarithmic space ☞ [Elberfeld, Jakoby, Tantau 10]
- ...thus in $\mathbf{AC}^1$
- ...thus in $\mathbf{IND}(\log n)$

- Therefore muddling allows us, in principle, to compute a tree decomposition

Question

- Can we maintain it for $\mathcal{O}(\log n)$ change steps?
- Probably not
- But with a slightly outdated tree decomposition we can muddle through for $\mathcal{O}(\log n)$ many changes

3-COL on btw-graphs: Illustration



- Phase 1&2: $\frac{1}{2} \log n$ steps
- Phase 3: $\frac{1}{2} \log n$ steps
- Phase 4: 1 step

☞ $\ell = 1 + \log n$

3-COL on btw-graphs: More detail (1/2)

- Compute colourability information for all *triangles* of the decomposition

Triangle

- Three bags B_1, B_2, B_3
  - B_2 is in the subtree of B_1
  - B_3 is in the subtree of B_1
  - B_2 is no predecessor or descendant of B_3

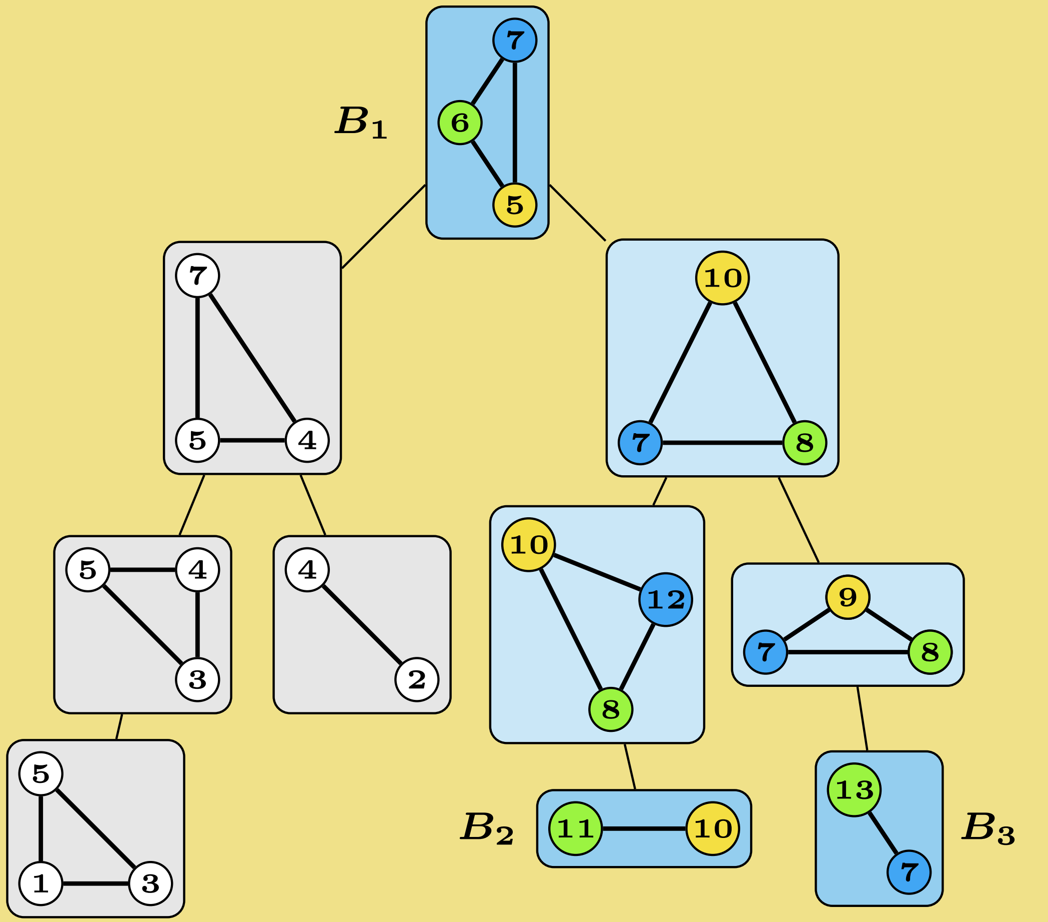
Boundary

- All nodes in B_1, B_2, B_3

Basic Idea

- Maintain all colourings of boundaries of triangles that can be extended to valid 3-colourings of the inner part of the induced graph

☞ slightly simplified



3-COL **on btw-graphs: More detail (2/2)**

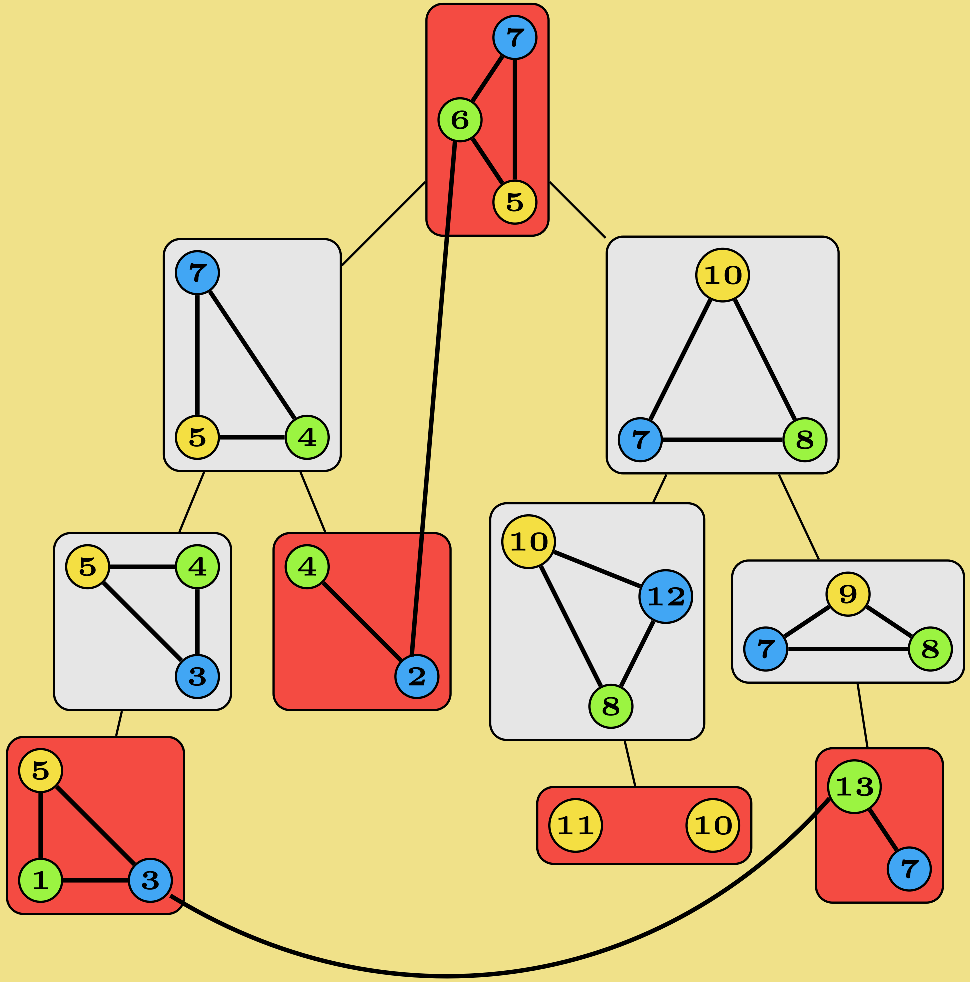
- If v is affected by a change: declare one bag of v as **special** ☞ special nodes

- After $\log n$ changes: $\mathcal{O}(\log n)$ nodes are special

- Existentially quantify colouring C of special nodes
  → **MSO** on subgraph with $\mathcal{O}(\log n)$ nodes
  ☞ to be addressed later

- Check: C is a valid 3-colouring of the graph induced by the special nodes

- Use auxiliary relations to check that C can be extended for the whole graph



Theorem [Datta, Mukherjee, S., Vortmeier, Zeume 17]

- Every MSO-definable query can be maintained in **DynFO** on graph classes of bounded tree-width

Contents

Towards a more Practical Setting

- Allow more complex changes
- Consider overall work

Complex Changes

- So far we only considered very simple change operations:
  - Insertion or deletion of a single tuple
  - No change of the universe/domain
- What about other kinds of changes?

“Arbitrary Changes”?

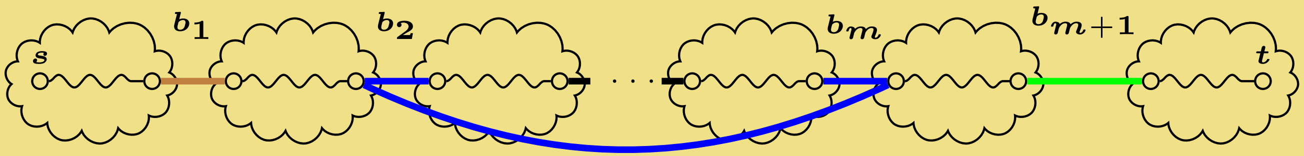
- If the database can change arbitrarily in one step, only **FO**-properties can be maintained in **DynFO**
- We consider two kinds of “non-arbitrary” complex changes
- ... with a limited number of changed tuples
- ... complex changes that are *defined* by formulas ☞ core SQL updates

Undirected Reachability under Complex Changes

Theorem [S., Vortmeier, Zeume 17]

- Reachability is in **DynFO** for undirected graphs in the presence of
  - single-tuple insertions and deletions and
  - **FO**-defined insertions

- Technique relies on a "bridge bound"
- Builds on spanning tree approach



- In a nutshell each sufficiently long path between connected components has a shortcut

- In general, the "bridge bound" can be non-elementary
- But for insertions defined by certain simple formulas (unions of conjunctive queries, UCQs), the number of bridges is small ☞ $2\times$ #-of-CQs

➞ Prototypical implementation works quite well

Reachability under $\log n$ insertions (1/2)

Theorem [Vortmeier, Zeume 19 (unpublished)]

- Reachability is in **DynFO*** under $\log n$ insertions

☞ $*$: If formulas can use $+$ and $\times$

Proof idea

(1) Compute Reachability for the $\log n$ affected nodes, and

(2) Combine this information with the Reachability information for the rest of the graph

Idea for (1)

- Reachability between two nodes x, y can be expressed by a monadic second-order (MSO) formula:

$$\forall X \; \big(X(x) \wedge \forall v \forall w \big(X(v) \wedge E(v,w) \rightarrow X(w)\big) \rightarrow X(y)\big)$$

Insight

- In our context, quantification of X is a-priori restricted to the subset W of affected nodes of size $\log n$

➨ The second-order $\exists X$ quantification ☞ restricted to W! can be replaced by a first-order quantification $\exists x$

☞ over all nodes!

Because:

- one node of the graph carries $\log n$ bits of information
- and this information can be decoded with the help of $+$ and $\times$

Reachability under $\log n$ insertions (1/2)

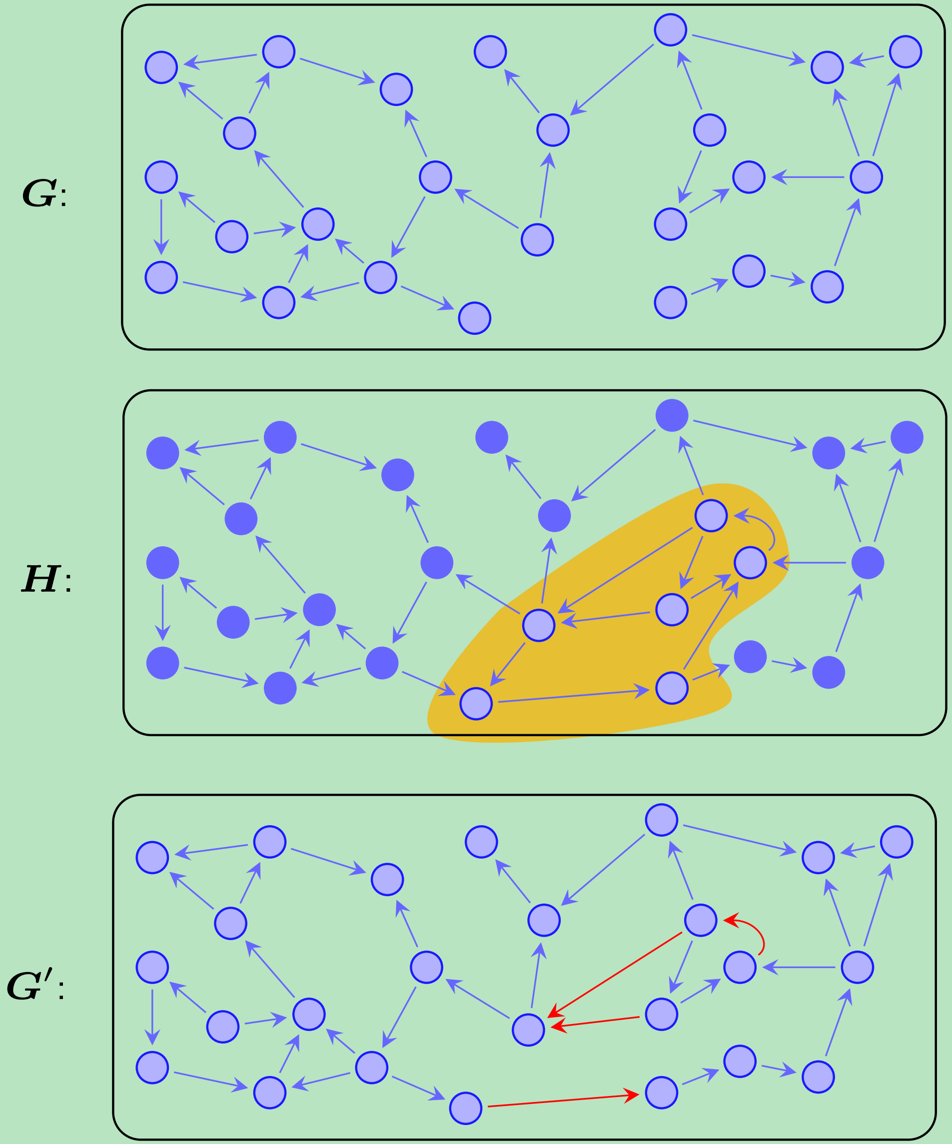
Proof idea (cont.)

- Assume that the transitive closure T_G of graph G is given

- After insertion of $\log n$ nodes
- ... let H be the graph induced by the affected nodes in the resulting graph G' ...
- ... with the newly inserted edges
- ... and additional edges for paths in G

- Compute transitive closure T_H of H
- Combine T_H with T_G to get $T_{G'}$

Proof idea (cont.)



Contents

Work-efficiency

- $q \in$ **DynFO** $\Rightarrow$
  q can be maintained in time $\mathcal{O}(1)$ on a PRAM

- That sounds quite efficient?
- But is it?

- An important quantity of a parallel algorithm is its **work**, i.e., the overall number of steps of all processors

Work of a dynamic program

- Rough upper bound: $\mathcal{O}(n^{a+d})$
  - a: arity of auxiliary relations
  - d: quantifier depth of update formulas

Question

- Which queries can be maintained in a (reasonably) work-efficient way?

- First study: word membership queries for formal languages

Dynamic complexity of formal languages: setting

Definition

- A **word structure** W representing a string w consists of
  - a set of positions $\{1, \ldots, n\}$,
  - with an ordering $<$,
  - and one unary relation S_σ for each alphabet symbol σ.

- For every i, there is at most one σ with $i \in S_\sigma$
  - In that case: $W_i \stackrel{\text{def}}{=} \sigma$
  - Otherwise: $W_i \stackrel{\text{def}}{=} \epsilon$

- The word represented by W is $W_1 \cdots W_n$

Example

- The word structures
b		b	c		c

    and
b	b	c		c	

  both represent the same string $bbcc$

- We allow two kinds of update operations:
  - operation **ins**$_\sigma(i)$, inserts i in $S_\sigma(i)$ and deletes it from all $S_{\sigma'}(i)$, for $\sigma' \neq \sigma$,
  - operation **del**(i), setting all $S_\sigma(i)$ to false.
- The linear order can not be changed!

Example

b	c	b			c

- **ins**$_c(2)$, **del**(4), **ins**$_b(6)$

Dynamic Complexity of Formal Languages

Definition

- **DynProp:**
  - Queries that can be maintained in **DynFO** with quantifier-free formulas and aux **relations**
- **DynQF:**
  - Queries that can be maintained in **DynFO** with quantifier-free formulas and aux **functions** (and relations)

✎ **DynQF** formulas can use "if-then-else"-terms

Theorem [Patnaik, Immerman 94/97]

- **Reg** $\subseteq$ **DynFO**
- All Dyck languages can be maintained in **DynFO**

Theorem [Hesse 03]

- **Reg** $\subseteq$ **DynQF**

Theorem [Gelade, Marquardt, TS 09/12]

- With respect to formal languages: **DynProp** $=$ **Reg**

Theorem [Gelade, Marquardt, TS 09/12]

- **CFL** $\subseteq$ **DynFO**
- All Dyck languages can be maintained in **DynQF**

Corollary

- **DynProp** $\subsetneq$ **DynQF**

Reg ⊆ DynProp

Proof idea

- Let $\mathcal{A} = (\Sigma, Q, \delta, s, F)$ be a DFA for the regular language L
- Maintain $R_{p,q}(i,j) \equiv \delta^*(p, w_{i+1} \cdots w_{j-1}) = q$

Example

w_1	$\ldots$	w_i	w_{i+1}	$\ldots$	w_{k-1}		w_{k+1}	$\ldots$	w_{j-1}	w_j	$\ldots$	w_n
			p			p'	p'			q		

Proof sketch (cont.)

- **on insert k into S_σ update $R_{p,q}(i,j)$ as**

$$\Big[(i<k<j) \wedge \bigvee_{\delta(p',\sigma)=q'} (R_{p,p'}(i,k) \wedge R_{q',q}(k,j))\Big] \vee \cdots$$

- **on delete k from S_σ update $R_{p,q}(i,j)$ as**

$$\Big[(i<k<j) \wedge \bigvee_{p'} (R_{p,p'}(i,k) \wedge R_{p',q}(k,j)\Big] \vee \cdots$$

- Work: $\theta(n^2)$

First Results (FoSSaCS 2021)

- Introduction of PRAM-like syntax for DynFO programs
- Upper bound results for maintaining formal language membership queries
-

Language class	Sequential time	DynFO work
FO definable	$\mathcal{O}(\log \log n)$	$\mathcal{O}(\log n)$
Regular	$\mathcal{O}(\frac{\log n}{\log \log n})$	$\mathcal{O}(n^{\epsilon})$ for $\epsilon > 0$
Dyck_1	$\mathcal{O}(\log n \log \log n)$	$\mathcal{O}(\log^3 n)$
Dyck_k	$\mathcal{O}(\log^3 n \log^* n)$	$\mathcal{O}(n \log^3(n))$

  (Sequential update time bounds: [Frandsen et al. 95/97])
- Currently: graph queries

Contents

Lower bounds: a sad state

Easy observation

- $q \in$ **DynFO** $\Rightarrow q \in$ **PTIME**
  - Just insert the tuples of $\mathcal{D}$ into an empty database one by one, and compute all updates

- So far there are no other general lower bound results for **DynFO**
- We cannot rule out that: **DynFO** $=$ **P**

- Most existing lower bounds apply to
  - auxiliary relations of bounded arity or
  - restricted logics or
  - both...

Reachability is not in unary DynFO (1/2)

Theorem [Dong Su 95/98]

- REACH $\notin$ unary **DynFO**

- unary **DynFO**: Update programs with unary auxiliary relations

Proof sketch

- Proof by contradiction with a locality argument
- Assume there is a unary dynamic program for REACH with m unary aux relations and a rule

  on delete (u, v) **from** E
  update $Q(x, y)$ **as** $\varphi(u, v, x, y)$

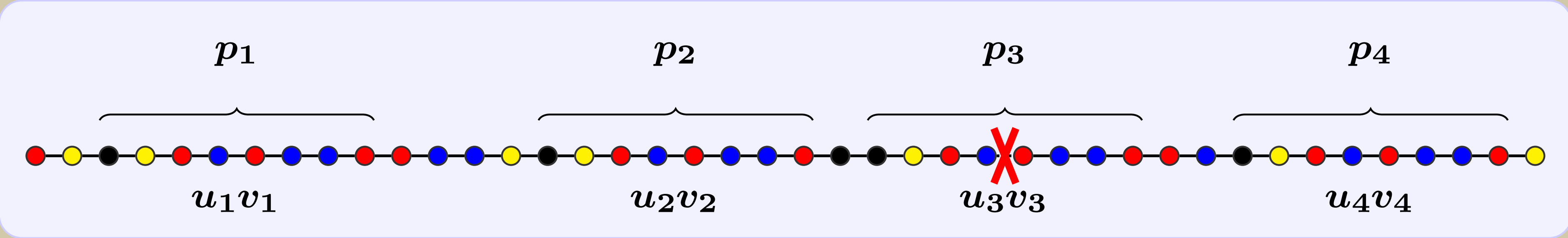
  with φ of quantifier-depth k

- The aux relations induce, for each node, one of 2^m colours

- Consider a graph consisting of a sufficiently long path with $\geqslant 4(2 \cdot 4^k + 2)2^{m(2 \cdot 4^k + 2)}$ nodes

Reachability is not in unary DynFO (2/2)

Example



Proof sketch (cont.)

- Since the path is long enough, there must exist four disjoint subpaths of length $2 \cdot 4^k + 2$ each with identical color (relations) sequence

- Let $(u_1, v_1), \ldots, (u_4, v_4)$ be the innermost edges of these subpaths
- After deletion of (u_3, v_3),
  - u_2 is still reachable from v_1, but
  - u_4 is no longer reachable from v_1

- The 4^k-neighborhoods of (v_1, u_3, v_3, u_2) and (v_1, u_3, v_3, u_4) are isomorphic

➨ $\varphi(u_3, v_3, v_1, u_2) \equiv \varphi(u_3, v_3, v_1, u_4)$ by Gaifman's Theorem

➨ After deletion of (u_3, v_3), the program gives the same answer for (v_1, u_2) and (v_1, u_4)

➨ The program is wrong with respect to either (v_1, u_2) or (v_1, u_4), the desired contradiction

Dynamic programs with quantifier-free formulas

- Hesse initiated the study of dynamic programs with quantifier-free update formulas [Hesse 03]

Definition

- **DynProp:**
  - Queries that can be maintained in **DynFO** with quantifier-free formulas and aux **relations**
- **DynQF:**
  - Queries that can be maintained in **DynFO** with quantifier-free formulas and aux **functions** (and relations)

✎ **DynQF** formulas can use "if-then-else"-terms

- Quantifier-free update formulas? Isn't that extremely weak?

Theorem [Hesse 03]

- Reachability is in **DynProp** for deterministic graphs ☞no quantifiers, aux relations

Theorem [Hesse 03]

- Reachability is in **DynQF** for undirected graphs ☞ no quantifiers, unary aux functions & relations

Some Further Inexpressibility Results

Theorem [Gelade, Marquardt, Schwentick 08/12]

- Alternating Reachability $\notin$ **DynProp**
- **FO** $\nsubseteq$ **DynProp**

Theorem [Zeume, Schwentick 13]

- REACH $\notin$ binary **DynProp**

Theorem [Zeume 14]

- If only edge insertions are allowed:
  - $\boldsymbol{k}$-CLIQUE can be maintained in $(\boldsymbol{k}-\mathbf{1})$-ary **DynProp**
  - $\boldsymbol{k}$-CLIQUE $\notin$ $(\boldsymbol{k}-\mathbf{2})$-ary **DynProp**

Contents

Conclusion

- **DynFO** is far more powerful than expected
- Upper bound results might be even "practical"

- Lower bounds for **DynFO** seem hopeless

- A lot remains to be done
  - Applications of the Reachability result
  - Implementations
  - Further exploration of linear algebra approaches
  - ...

References (1/2)

- Guozhu Dong and Jianwen Su. First-order incremental evaluation of datalog queries. In *DBPL 1993*, pages 295–308, 1993
- Sushant Patnaik and Neil Immerman. Dyn-FO: A parallel, dynamic complexity class. In *PODS 1994*, pages 210–221, 1994
- Sushant Patnaik and Neil Immerman. Dyn-FO: A parallel, dynamic complexity class. *J. Comput. Syst. Sci.*, 55(2):199–209, 1997
- Guozhu Dong and Jianwen Su. Arity bounds in first-order incremental evaluation and definition of polynomial time database queries. *J. Comput. Syst. Sci.*, 57(3):289–308, 1998
- D. A. M. Barrington, N. Immerman, and H. Straubing. On uniformity within NC^1. *Journal of Computer and System Sciences*, 41:274–306, 1990
- Kousha Etessami. Dynamic tree isomorphism via first-order updates. In *PODS*, pages 235–243. ACM Press, 1998
- William Hesse. The dynamic complexity of transitive closure is in $DynTC^0$. In *ICDT 2001*, pages 234–247, 2001
- Bastian Laubner. *The structure of graphs and new logics for the characterization of Polynomial Time*. PhD thesis, Humboldt University of Berlin, 2011
- Gudmund Skovbjerg Frandsen and Peter Frands Frandsen. Dynamic matrix rank. *Theor. Comput. Sci.*, 410(41):4085–4093, 2009
- Samir Datta, William Hesse, and Raghav Kulkarni. Dynamic complexity of directed reachability and other problems. In *ICALP (1)*, pages 356–367, 2014

References (2/2)

- Samir Datta, Anish Mukherjee, Nils Vortmeier, and Thomas Zeume. Reachability and distances under multiple changes. In *45th International Colloquium on Automata, Languages, and Programming, ICALP 2018, July 9-13, 2018, Prague, Czech Republic*, pages 120:1–120:14, 2018
- Samir Datta, Anish Mukherjee, Thomas Schwentick, Nils Vortmeier, and Thomas Zeume. A strategy for dynamic programs: Start over and muddle through. In *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017, July 10-14, 2017, Warsaw, Poland*, pages 98:1–98:14, 2017
- Thomas Schwentick, Nils Vortmeier, and Thomas Zeume. Dynamic complexity under definable changes. In *20th International Conference on Database Theory, ICDT 2017*, pages 19:1–19:18, 2017
- Samir Datta, Raghav Kulkarni, Anish Mukherjee, Thomas Schwentick, and Thomas Zeume. Reachability is in dynfo. In *International Colloquium on Automata, Languages, and Programming, ICALP 2015, Proceedings, Part II*, pages 159–170, 2015
- Thomas Zeume and Thomas Schwentick. Dynamic conjunctive queries. In *Proc. 17th International Conference on Database Theory (ICDT), Athens, Greece, March 24-28, 2014*, pages 38–49, 2014
- Thomas Zeume and Thomas Schwentick. On the quantifier-free dynamic complexity of reachability. In *Mathematical Foundations of Computer Science 2013*, pages 837–848, 2013
- Wouter Gelade, Marcel Marquardt, and Thomas Schwentick. The dynamic complexity of formal languages. *ACM Trans. Comput. Log.*, 13(3):19, 2012